Comparing bit values

A byte comprises eight bits that can each contain a **1** or a **0** to store a binary number representing decimal values from 0 to 255. Each bit contributes a decimal component only when that bit contains a **1**. Components are designated right-to-left from the "Least Significant Bit" (LSB) to the "Most Significant Bit" (MSB). The binary number in the bit pattern below represents decimal 50.

Bit No.	8 MSB	7	6	5	4	3	2	1 LSB
Decimal	128	64	32	16	8	4	2	1
Binary	0	0	1	1	0	0	1	0

Although the **char** data type is the basic 1-byte storage unit in C programming, as described on the previous page, it is possible to manipulate individual parts of a byte using "bitwise" operators.

Operator:	Name:	Binary number operation:
|	OR	Return a **1** in each bit where either of two compared bits is a **1** Example: **1010 | 0101 = 1111**
&	AND	Return a **1** in each bit where both of two compared bits is a **1** Example: **1010 & 1100 = 1000**
~	NOT	Return a **1** in each bit where the bit is not **1**, and return **0** where the bit is **1** Example: **~ 1010 = 0101**
^	XOR	Return a **1** in each bit where only one of two compared bits is a **1** Example: **1010 ^ 0100 = 1110**
<<	Shift left	Move each bit that is a **1** a specified number of bits to the left Example: **0010 << 2 = 1000**
>>	Shift right	Move each bit that is a **1** a specified number of bits to the right Example: **1000 >> 2 = 0010**

Don't forget

Many C programmers never use bitwise operators but it is useful to understand what they are and how they may be used.

Hot tip

Each half of a byte is known as a "nibble" (4 bits). The binary numbers in the examples in the table describe values stored in a nibble.